iPhone OS Audio

Session 315

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

iPhone OS Audio

William Stewart
Digital Audio Plumber

iPhone OS Audio

- APIs are available for playing and recording audio
- AudioSession and Audio Behaviour
  - Category
  - Routes
  - Interruptions
  - Managing the state of the audio system

Audio APIs for Play and Record

Audio Play and Record APIs

- MPMusicPlayerController
- AVAudioPlayer
- AVAudioRecorder
- AudioQueue
  - Out/In
- OpenAL
- AudioUnits (AURemoteIO)
  - Out/In

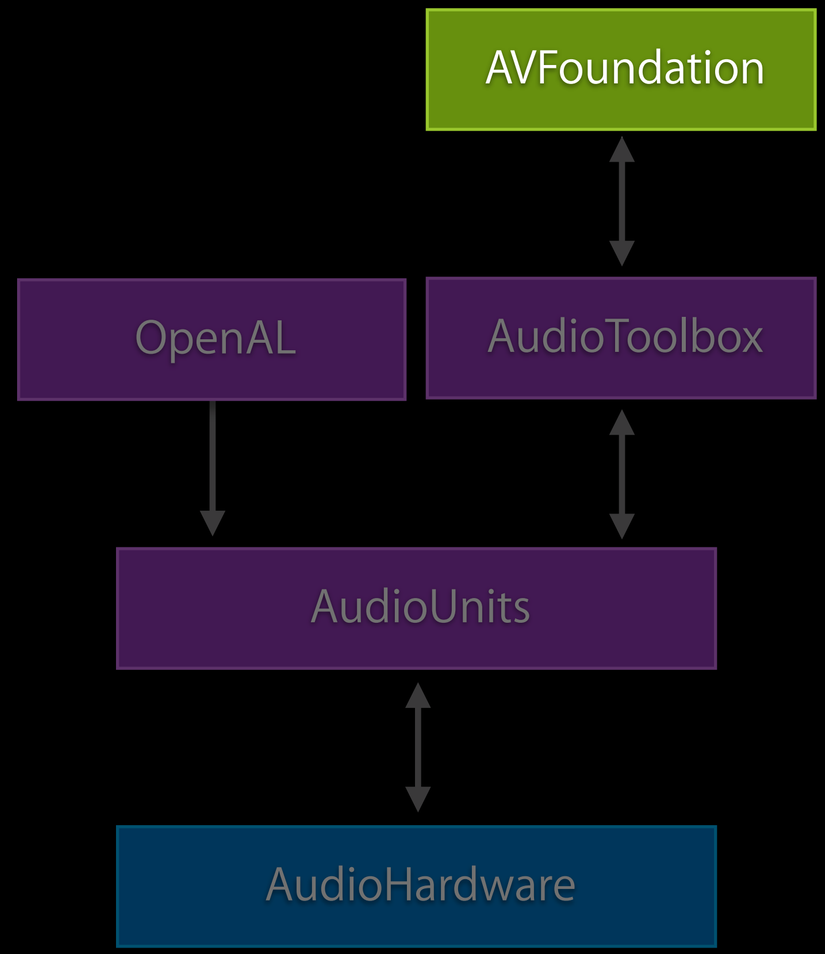
iPod Library Access

- MediaPlayer Framework
  - MPMusicPlayerController
    - iPod Library Access
    - Plays with the same rules as iPod

AVFoundation

AVAudioPlayer/AVAudioRecorder

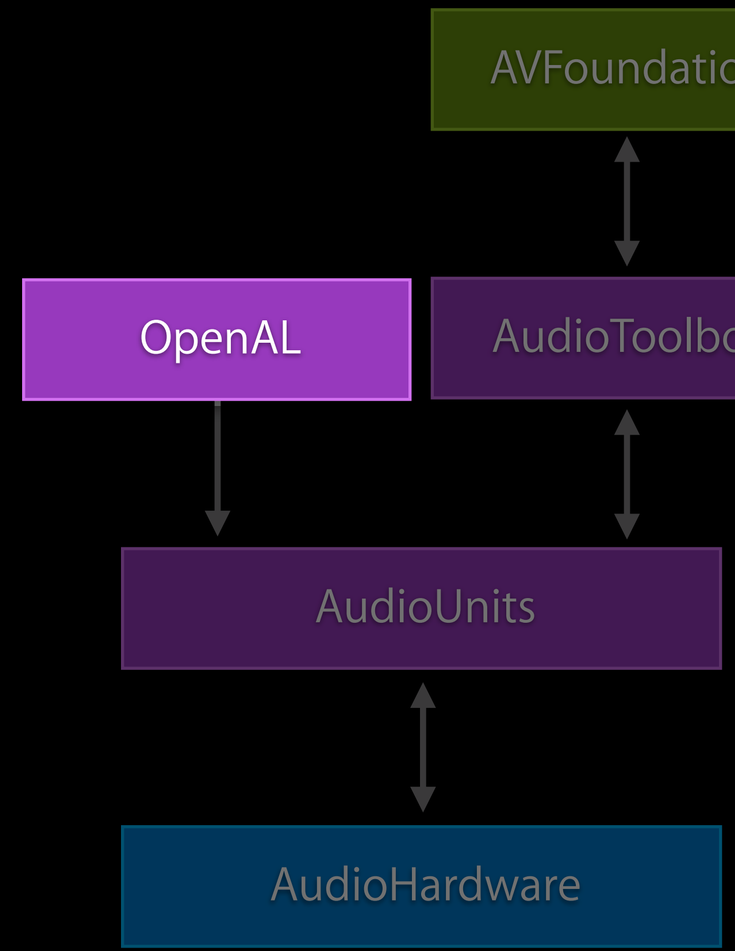
- Built on AudioQueue and AudioFile APIs
- Simplest API to use for playing and recording files
- Usage is utilitarian
  - Oriented around task
  - Will handle active state
    - Default category
    - Reactivate on end interruption
  - Self-contained solution (with delegates)



OpenAL

3D audio game engine

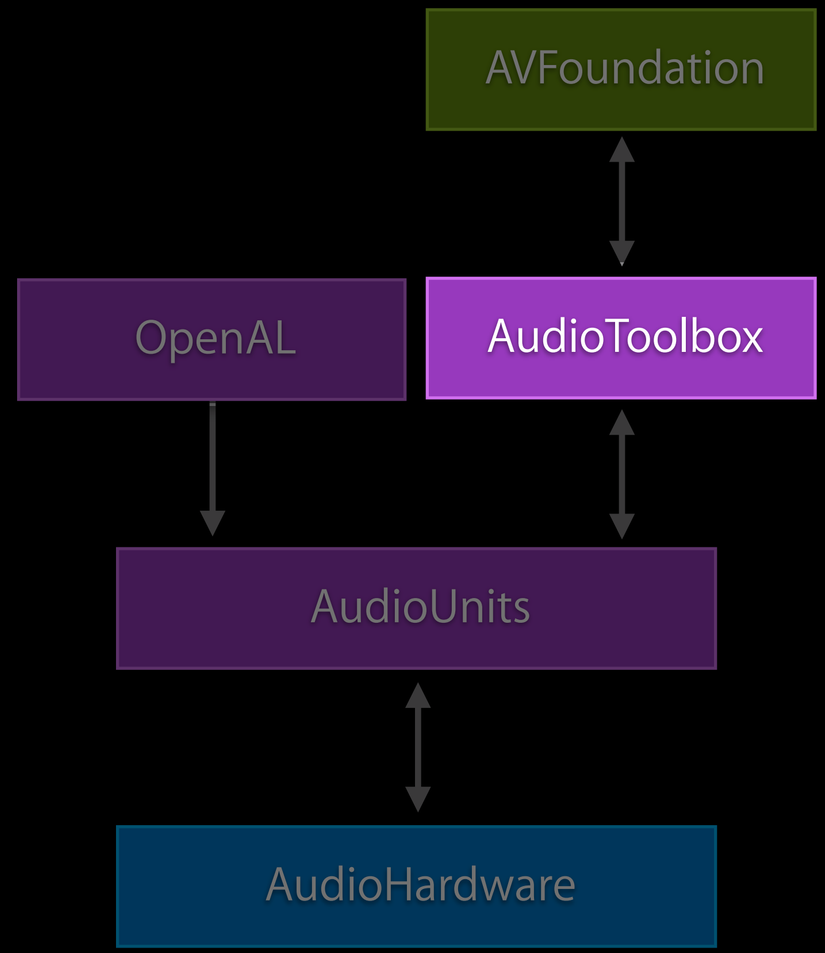
- Localised source positioning
- Independent input rate control
- “Doppler” effect



AudioToolbox

General audio services

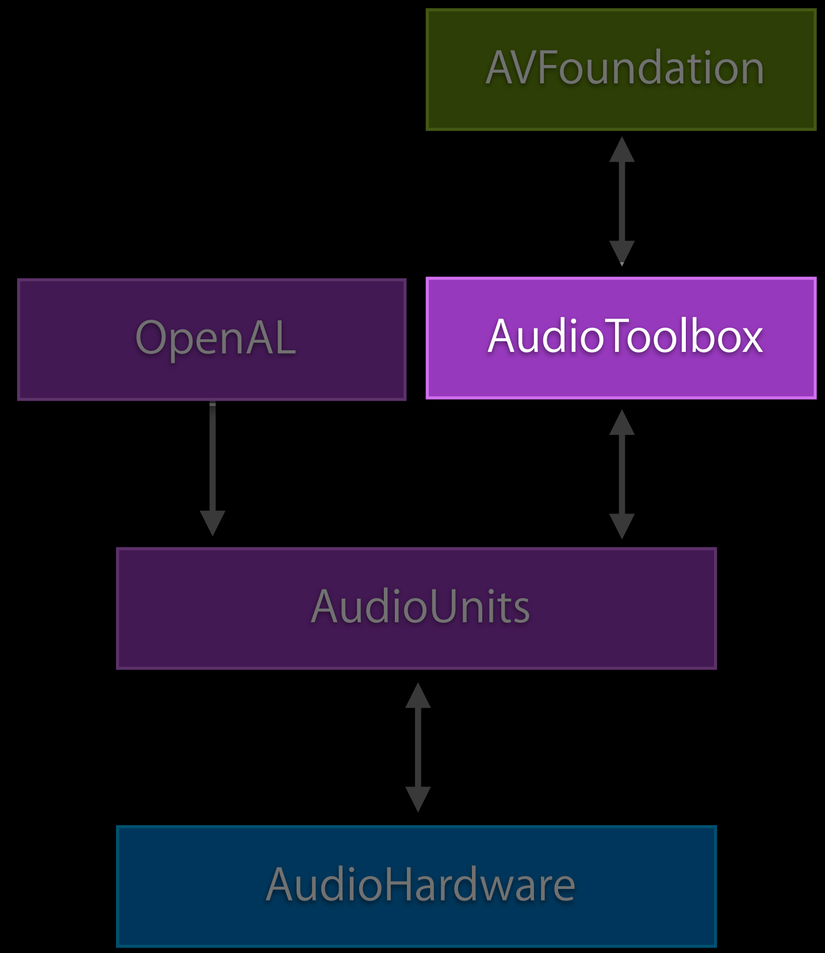
- AudioQueue
  - Play and record
- AudioFile
  - Read and write files



AudioToolbox

General audio services

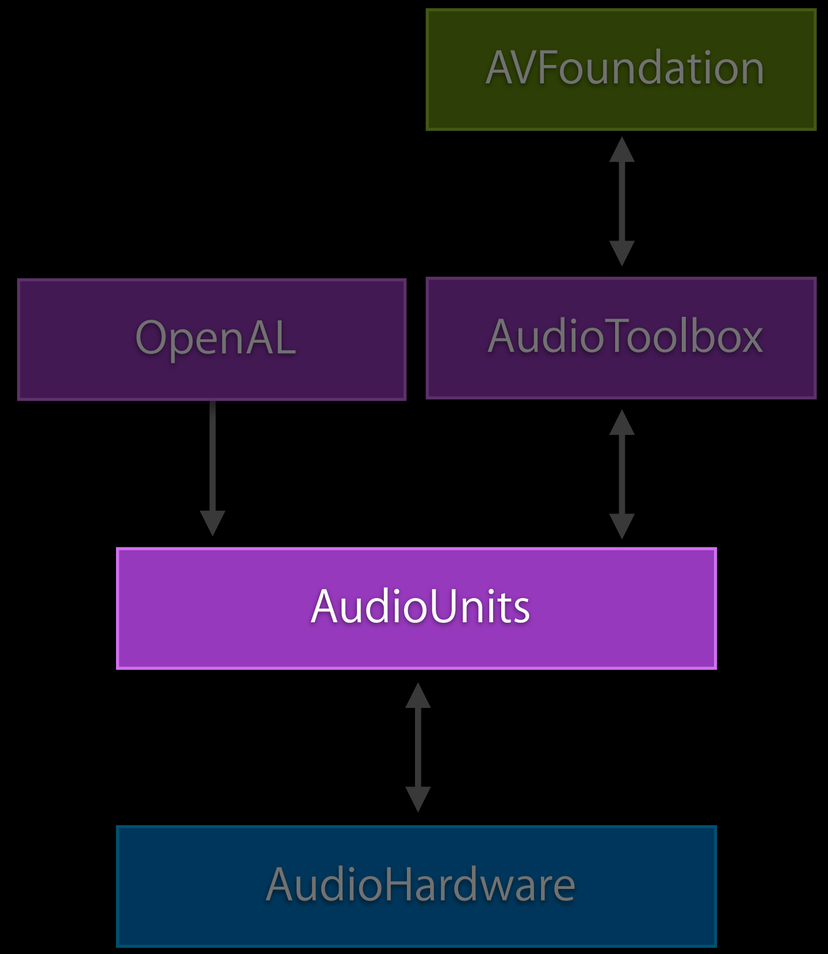
- Audio Converter
  - Convert audio formats
  - Access to audio codecs
- ExtendedAudioFile
  - Combines AudioFile and AudioConverter
- Audio Format
  - Manage audio formats
- Audio Services Play
  - System and Alert sounds



AudioUnit

Low-level audio services

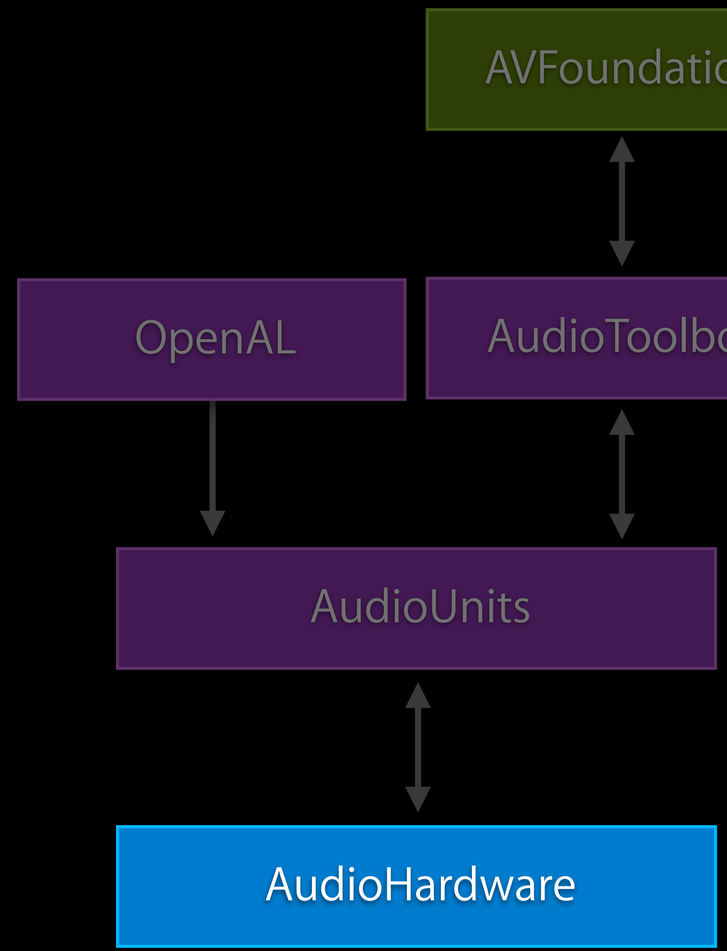
- AURemoteIO
  - Base level audio I/O
- AUVoiceProcessing
  - Adds voice processing operations to I/O
- Mixer audio units



Audio Hardware

Audio I/O and hardware acceleration

- No direct access
  - Access through higher level API
- Manage state through Audio Session



Using Audio Session

AudioSession

- Application's interaction with the audio system
- Current state of audio on device
  - Used to request preferred audio device state
  - Where is audio going to/coming from
- State transitions
  - Explicit notion of "being active"
  - Explicit notion of "interruption"
    - A higher priority item (for example, a ring-tone from a phone call) needs to interrupt currently playing audio

AudioSession API

`<AudioToolbox/AudioServices.h>`

- AudioSession APIs
- C
- All of the implementation/detail is here

AudioSession API

`<AVFoundation/AVAudioSession.h>`

- Convenience wrapper for most critical functionality
- ObjC

- APIs refer to the same underlying, implementation

AudioSession Categories

- Basic role/services available to an application
- Establish an application's priority
  - Whether to allow mixing with iPod playback
  - Does it interrupt currently playing apps
- Establish fundamental behaviour
  - Ringer switch
  - Screen lock
  - Input/output audio routes
    - For example: Speaker or receiver?

Using AVAudioSession

```
AVAudioSession session = [AVAudioSession sharedInstance];

[session setCategory: AVAudioSessionCategoryPlayback
               error: nil];
```

AudioSession Categories

- Primary features for categories
  - Mix with others
  - Obey ringer switch
  - Go silent when phone locks
  - Has input
  - Has output

Category Behaviour

	Obey Ringer Switch	Obey Screen Lock	Mix with Others	Audio Input	Audio Output
Ambient	●	●	●		●
SoloAmbient	●	●			●
Playback			Override		●
Record				●	
PlayAndRecord			Override	●	●

Audio Category Examples

- Ambient
  - Game (game-play sounds)
- AmbientSolo
  - Game providing its own background music
- Playback
  - iPod, YouTube/Safari media playback
- PlayAndRecord
  - Chat
    - Both input and output enabled
    - Don’t need to do both at the same time though!
- Record
  - Record application
    - Just input (so no UI sounds will play)

Deprecated

- UI Sounds == Ambient
- Live Audio == Playback

Customisable Category Behaviour

`kAudioSessionProperty_OtherAudioIsPlaying`

- Other audio is playing
  - Game should play background music or not
  - May use this as a decision point to decide on what category to specify

Customisable Category Behaviour

`kAudioSessionProperty_OtherMixableAudioShouldDuck`

- Other audio is attenuated when application plays audio
  - GPS
  - Nike App

Demo

“Mix with Others” Characteristic

- Can a background app play/record while your app is active
  - iPod playback
  - MPMusicPlayerController invoked with your application
- This is about whether “your” app interrupts “another” app

“Mix with Others” Characteristic

- Hardware and software audio codecs
  - Hardware
    - Low CPU cost, low power usage
  - Software
    - Flexibility of use
    - Mixing with iPod
    - More CPU Cost than H/W version
      - Can affect application performance

Hardware Supported Audio Formats

- Audio decoders
  - MPEG-4 AAC
  - MP3
  - Apple Lossless (ALAC)
  - MPEG-4 HE-AAC
    - New with 2.2
- Audio Encoders
  - MPEG-4 AAC
    - iPhone 3G S only

Software Decoders

- New with 3.0
  - MPEG-4 AAC
  - MP3
  - Apple Lossless (ALAC)
- All of the other formats
  - ulaw/alaw
  - iLBC
    - Speech codec
  - …

Access to Hardware Codecs

- Must have “mix with others” set to false
  - Ambient Solo
  - Playback
  - PlayAndRecord
  - Record (for AAC encoding)
- Overriding this setting also removes access to h/w versions

	Mix with others
Ambient	●
SoloAmbient	
Playback	●
Record	
PlayAndRecord	●

Using Hardware Codecs

- See <AudioToolbox/AudioFormat.h>
- Can explicitly control use of hardware or software with AudioQueue
- If you use a format supported in hardware, but have “mix with others” set to true
  - Default to software codec

	Mix with others
Ambient	●
SoloAmbient	
Playback	Override
Record	
PlayAndRecord	Override

Customisable Category Behaviour

`kAudioSessionProperty_OverrideCategoryMixWithOthers`

- Use with AudioSessionSetProperty
  - Defaults to false
  - Application must be prepared for setting this property to fail as behaviour may change in future releases
  - If your application changes its category, you should reassert the override
  - It is not sticky across category changes

AudioSession Routes

- Where does the audio go?
- Where does the audio come from?

Audio In, Out and Controls

Bluetooth™

A2DP Headphones/Speaker

Receiver

Bluetooth™

Headphones

Speaker

Microphone

Headset

Audio In, Out and Controls

Audio Routes

- Based on user state “last in wins”
- Audio Device Picker
  - Wireless devices cannot be “jacked in”

Audio Device Picker

Audio Route

`kAudioSessionProperty_AudioRoute`

- The current route

`kAudioSessionProperty_AudioRouteChange`

- Notification for when the route changes
- Reason why route changed
- What was the old route

What Is the Current Route?

```
CFStringRef route;
UInt32 ioSize = sizeof(route);
AudioSessionGetProperty (
    kAudioSessionProperty_AudioRoute,
    &ioSize,
    &route);
```

- Route
  - Describes current route
  - “” for no current route

Audio Route Changes

- Register for notifications for when a route changes

```
AudioSessionAddPropertyListener(
    kAudioSessionProperty_AudioRouteChange,
    MyPropListener,
    &myClientData);
```

Audio Route Change Notification

```
void MyPropListener (void* inClientData, AudioSessionPropertyID inProp,
                     UInt32 dataSize, void* data)
{
   if (inProp == kAudioSessionProperty_AudioRouteChange) {
      CFDictionaryRef routeChange = (CFDictionaryRef)data;
        routeChange[reason] -> { number }
          why did the route change?
            new device
            existing device went away
        routeChange[old_route] -> { string }
   }
}
```

Audio Routes

`kAudioSessionProperty_OverrideAudioRoute`

- Used with `AudioSessionSetProperty`
  - Set to “speaker”
    - Switches current audio destination to the speaker
  - Set to “none”
    - Go back to the “last in” rule
- Typical use case is speaker phone
- Overrides are not sticky
  - As route changes, override setting is lost and would need to be reasserted

Demo

Audio Hardware Format

- Get the Current State of audio hardware
  - Sample Rate (In/Out)
  - Num Channels (In/Out)

Audio Hardware Format

- AVAudioSession

```
[session currentHardwareSampleRate]
[session currentHardwareInputNumberOfChannels]
[session currentHardwareOutputNumberOfChannels]
```

Preferred Hardware Settings

- Establish your preferred settings for when you assert your use of the audio system
  - Sample Rate
  - Num Channels
  - I/O Size
- These are requested settings, but the hardware may not be set to these settings

Preferred Audio Hardware Settings

- AVAudioSession

```
[session setPreferredHardwareSampleRate: srate
    error: nil]

[session setPreferredIOBufferDuration: duration
    error: nil]
```

Audio Hardware Volume

- Volume is set
  - Per current categories, per route
    - For example, there is a volume for:
      - “headphones: phone call”
      - “speaker: playback”
    - etc.
- Not all routes have volumes
  - A2DP has no volume setting

Audio Hardware Volume

- To get the current volume

```
Float32 volume;
UInt32 ioSize = sizeof(volume);
AudioSessionGetProperty(
    kAudioSessionProperty_CurrentHardwareOutputVolume,
    &ioSize,
    &volume);
```

- To set
  - UIKit volume sliders to present a UI
  - User sets with volume keys

Audio Hardware Properties

- Has input

```
UInt32 hasInput;
UInt32 size = sizeof(hasInput);
AudioSessionGetProperty(
    kAudioSessionProperty_AudioInputAvailable,
    &hasInput,
    &size);
```

- Instantiate a property listener for when this changes
  - iPod
    - Insert or remove headset with Mic

Demo

Audio Session Management

- What are the typical use cases for audio session
  - Initialisation
  - Establish basic state
    - Preferred hardware state
  - Set active
  - Respond
    - To changes of state
    - To interruptions

Using AVAudioSession

- With AVAudioSession initialisation is implicit

```
AVAudioSession session = [AVAudioSession sharedInstance];
```

- Implement delegate methods to deal with notifications of state changes

```
session.delegate = MyDelegate;
```

  - Interruption
  - Change in hardware state

Using AVAudioSession

- Set appropriate category
  - Interaction (role, priority) with the audio system

```
[session setCategory: AVAudioSessionCategorySoloAmbient
               error: nil];
```

- Any other state you wish to establish
  - Hardware properties
  - Route
  - etc.

Active Audio Session

```
[session setActive: YES error: nil]
```

- Asserted category and behaviours that specifies
- Application now has use of the audio system
  - As described by category
- Application loses use of audio system if
  - Session is interrupted
  - Explicitly de-activates itself
  - Application quits

Active Audio Session

```
[session setActive: NO error: nil]
```

- If record category
  - Make your app inactive when not recording
  - Allows notification sounds to play

Changing Audio Session Active

- This is not a “change on play” type property
- Rather, an audio session is about personality
  - In playback app the session can be active all the time
  - In a record and play app, the session can be active all the time
  - In a record only app, the session should be active only when recording
- Can be a system wide interruption

AudioSession Priority and Interruptions

- Active Session
  - Decides priority based on category settings
    - Mix with others
    - Require input/output
- Your application can interrupt iPod playback if category appropriate
- Alarm/Phone Call interrupts any “active session”
  - Your app
  - iPod

AudioSession Interruptions

- Makes your session inactive
  - Currently playing audio is stopped
- You cannot become active again until the higher priority item is complete

AudioSession Interruptions

- `(void)beginInterruption`
  - Have no control over audio playback
    - Already inactive
  - Change state of UI, etc, to reflect non-playing
    - For instance, change play button to its stopped state
- `(void)endInterruption`
  - Make session active
  - Resume playback, etc
    - Whatever is appropriate for your app

Demo

Using AudioSession

- Use in conjunction with AVAudioSession
  - More advanced features
- Use the Get and Set property calls to manage state
- Instantiate property listeners for the various properties that you are interested in
  - Route changes
  - Hardware changes
  - Other audio is playing
  - etc.

Summary

- Use AVAudioSession
- Manage application role
  - Category
- Interact with audio environment
  - Routes
  - Hardware states
  - Etc.
- Minimal extra work for better user experience

Related Sessions

Game Development for iPhone, Part 1	Presidio Tuesday 10:30AM
Audio Processing and Playback	Presidio Tuesday 5:00PM

Labs

Audio Lab	iPhone Lab B Wednesday 12:00PM
Audio Lab	Graphics Lab A Wednesday 3:30PM

More Information

Allan Schaffer
Graphics Technology Evangelist
aschaffer@apple.com

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09